Using Coordinates to Move a Graphic

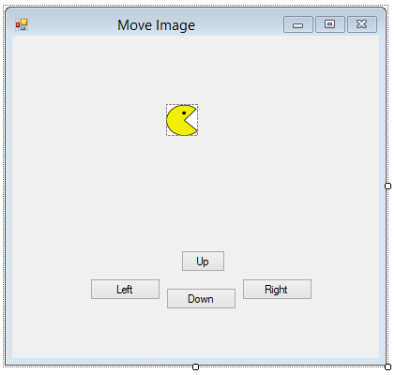
Step 1 – Create your form and user interface

We will start by creating a new project. It will be called Move Image. Create the following interface:

frmMoveImage **imgPacMan**

btnLeft **btnRight**

btnUp **btnDown**



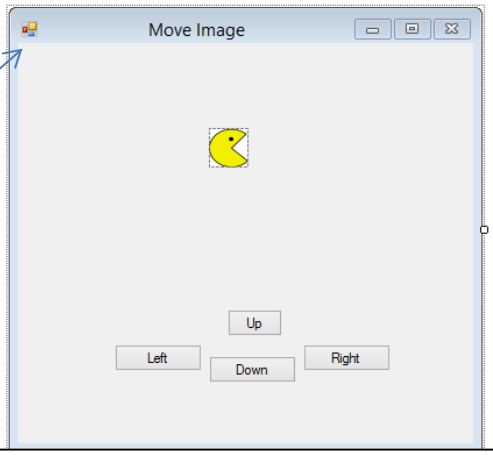
Create an integer variable called **intSpeed** and set it to 5.

Import the four images provided in the Images folder (or any images of your choice) into the project resources. These will represent your moving character. Add any of the images from resources to your Picture Box and set its **Sizemode** property to **Stretched**.

Notice your Picturebox (and many other controls) has a **Location** property that includes an X and Y coordinate. This is the property that we will change to alter the location of our character.

Step 2 – Moving the Character and the coordinate system

Because the upper-left portion of an application window tends to be fixed, this is where the coordinate (0, 0) is. The x-coordinates increase as you move right, and the y-coordinates increase as you move down.



Add the following line of code to the btnRight click event:

We can set the Location property of our PictureBox to a new location by making a new Point

We do not want to change the y-coordinate.

```
imgPacMan.Location = New Point(imgPacMan.Location.X + intSpeed, imgPacMan.Location.Y)
```

imgItem.Location.X will give us the x-coordinate of the upper left corner of **imgItem**.

Here we add **intSpeed** which has a value of 5 to the current x-coordinate of our image.

Run your program.

How do I Keep my character from running of the side of the screen?

Each time we are about to change the location of our image we can test to see if it will hit the edge of our screen and if so, not change its location.

Add the following code to your **btnRight** click event:

Because the x-coordinate of the **Location** of our Picturebox is its upper **left** hand corner, we need to add its width and the speed so we know the coordinate of its right side after the move.

Me.Width is the width of the Form.

```
 If imgPacMan.Location.X + imgPacMan.Width + intSpeed < Me.Width Then
    imgPacMan.Location = New Point(imgPacMan.Location.X + intSpeed, imgPacMan.Location.Y)
Else
    imgPacMan.Location = New Point(Me.Width - imgPacMan.Width, imgPacMan.Location.Y)

 End If
```

If our Picturebox does go off the edge of the form when we move it, we can just place it at the edge instead.

Things to try:

- What effect does changing **intSpeed** have on your program?
- What could you add to allow the user the ability to change the speed of PacMan themselves?
- Add the code for the Up, Down and left buttons
- Make the user not able to move the image off your form in any direction.
- When the user clicks a direction button, have PacMan face in the appropriate direction (try to do this by only changing the image in the picturebox when a new direction is chosen not each time a button is clicked)
- Put in a background image
- When the player moves their character off the edge of the screen, have him enter from the opposite side

How can we have our character eat a cookie?

Create a Picturebox and name it **imgCookie**. Find the image of a cookie, and add it to the resources and then put that image into **imgCookie**.

Each time we move our character, we need to check to see if we collided with our cookie. Do this with the following code in the **btnRight** click event:

```
'Checks for collision with cookie
If imgPacMan.Bounds.IntersectsWith(imgCookie.Bounds) Then
            imgCookie.Visible = False
End If
```

This will return True if the rectangles that make up the boundaries of the two Pictureboxes touch.

How can I have my character be blocked by an obstacle?

Create a Picturebox and name it **imgRock**. Find the image of a rock, and add it to the resources and then put that image into **imgRock**.

Each time we move our character, we need to check to see if we collided with our wall. If we detect a collision, we will place intersecting the edges of our two image boxes next to one another. Do this with the following code in the **btnRight** click event:

This will return True if the rectangles that make up the boundaries of the two Pictureboxes touch.

```
'Checks for colission with wall
If imgPacMan.Bounds.IntersectsWith(imgRock.Bounds) Then
       imgPacMan.Location = New Point(imgRock.Location.X - imgPacMan.Width, imgPacMan.Location.Y)
End If
```

We want the right side of our character Picturebox to rest against the left side of the obstacle.

Things to try

- When your character eats the cookie, instead of making it invisible, change the image to crumbs
- Add multiple 'edible' objects
- Change the players speed/image/behavior in some way when an object is consumed
- Add walls
- Add a finish line

You may want to try the **Keyboard Events** tutorial before you start on this mini project, and then have the movement based on events from the keyboard instead of button clicks.

Mini Project

Create a mini maze game with your own characters and graphics. Add some interesting features such as:

- Have a finish line unlocked by a key that must be eaten
- Present some type of victory screen upon completion
- Have walls that change location to block or unblock sections when certain power-ups are consumed
- Have hazards that cause the player to 'die'
- Add a transporter
- Add a health mechanic
- Any other interesting ideas